Tight Upper Bounds for the Spectral Radius via Diagonal Similarity Scaling: Matmul-Free Coordinate Descent, Operator-Norm Menu, and Reproducible Single-PC Benchmarks

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

September 22, 2025

Sharp, inexpensive upper bounds on the spectral radius ρ(AB) matter across matrix analysis, control, and AI stability. Practitioners typically use classical surrogates such as $c_AB = \sqrt{(\|AB\|_1\|AB\|_\infty)}$ and $c_BA = \sqrt{(\|BA\|_1\|BA\|_\infty)}$. We introduce a simple diagonal similarity split that reliably tightens these bounds without heavy linear algebra: write $A' = D^{-1}A$ and $B' = BD$ with $D = \mathrm{diag}(e^x)$, so $\rho(AB) = \rho(A'B')$ but the induced $1/\infty$ norms of A', B' change. Our bound is the minimum of a small *operator-norm menu*: { c_AB, c_BA, $\|A'\|_1\|B'\|_1$, $\|A'\|_\infty\|B'\|_\infty$, $\sqrt{[(\|A'\|_1\|A'\|_\infty)(\|B'\|_1\|B'\|_\infty)]}$ }. We choose x by a matmul-free coordinate descent with step-halving; the loop costs $O(n^2)$ per sweep because norms update by column/row reweighting, while c_AB and c_BA are precomputed once. On structured families (Toeplitz decay, heterogeneous row-stochastic/Markov pairs, and banded diagonally dominant systems), we observe consistent tightening over c_AB: for Markov pairs, ratio_avg ≈ 0.914 (≈8.6% improvement) with winrate = 1.000; banded DD shows smaller but robust gains; Toeplitz provides a clear limit case (ratio_avg = 1.000). The artifact is fully reproducible on a single PC via short PowerShell entry points that generate runs, manifests, CSV exports, and auto-rendered tables—facilitating auditability and downstream plotting. For AI, the method offers fast, deterministic surrogates for stability screening (e.g., compounding linear layers or Markov operators); for mathematicians, it is a clean diagonal-similarity device that interacts predictably with induced norms and Perron–Frobenius structure.

Keywords: spectral radius, diagonal similarity, matrix scaling, operator norms, induced norms, AB/BA bounds, Perron–Frobenius, Markov operators, Toeplitz matrices, banded diagonally dominant matrices, Gershgorin-type reasoning, nonconvex optimization, coordinate descent, step halving, matrix analysis, numerical linear algebra, stability analysis, control, reproducibility, single-PC experiments, symbolic computation.

A collaboration with GPT-5. CC4.0.

Outline

1. **Problem & Contributions**
    - Need: sharper yet cheap upper bounds for $\rho(AB)$ relevant to matrix analysis, control, and AI stability.
    - Contributions: (i) diagonal-similarity menu bound; (ii) $O(n^2)$ matmul-free coordinate descent; (iii) sufficient conditions/limits; (iv) reproducible artifact and benchmarks.
2. **Background & Related Work**
    - Induced norms, submultiplicativity, classical c_AB / c_BA.
    - Diagonal scaling/balancing and similarity invariants; connections to Gershgorin and matrix equilibration; contrasts with heavy SDP/convex relaxations.
3. **Method**
    - Split: $A' = D^{-1}A$, $B' = BD$ with $D = \mathrm{diag}(e^x)$.
    - Menu bound: $\min\{ c_AB, c_BA, \|A'\|_1\|B'\|_1, \|A'\|_\infty\|B'\|_\infty, \sqrt{[(\|A'\|_1\|A'\|_\infty)(\|B'\|_1\|B'\|_\infty)]} \}$.
    - Fast updates: column-scaled 1-norms and row-scaled ∞-norms in $O(n^2)$.
    - Optimizer: coordinate line-search, step-halving, stopping, complexity.
4. **Theory Note**
    - Invariance under diagonal similarity; when heterogeneity enables strict tightening (row-stochastic/Perron–Frobenius settings).
    - Nonconvex landscape; coordinate-wise improvement guarantees; counterexamples (Toeplitz).
5. **Experimental Setup**
    - Families: Toeplitz decay ($r \in (0,1)$), heterogeneous Markov pairs, banded DD (bandwidth k, dominance α).
    - Metrics: winrate = $\Pr[B(A,B;x^*) < c_AB]$; ratios {avg, min, max} vs. c_AB.
    - Implementation details; runtime profile; no matmul in the loop.

6. **Results**
    - Markov: winrate = 1.000; ratio_avg ≈ 0.914.
    - Banded DD: winrate = 1.000; ratio_avg ≈ 0.977.
    - Toeplitz: ratio_avg = 1.000 (limit case).
    - Ablations: step sizes, iteration caps, dimension scaling.
7. **Implications for AI & Control**
    - Rapid stability screening for composed linear maps and Markov transitions; integration into verification pipelines; guidance on when to expect gains.
8. **Reproducibility & Artifact**
    - Scripts: tools/refresh.ps1, tools/make_tables.py, tools/export_csvs.py.
    - Runs/manifest structure; CSVs and docs/results_tables.md for plotting.
9. **Discussion & Future Directions**
    - Extending the menu (other matrix norms, block/weighted scalings).
    - Links to symbolic/typed pipelines (equational reasoning) and larger operator chains.
    - Open problems: tighter certificates, worst-case guarantees, multi-D scalings.
10. **Appendices**

- A: Pseudocode & complexity proofs.
- B: Full tables and per-trial logs.
- C: Proof sketches and counterexamples.

References

Problem & Contributions

The problem

For many pipelines in matrix analysis, control, and AI, one repeatedly needs a fast *upper bound* on the spectral radius ρ(AB)—e.g., for stability screens, step-size selection, or conservative certificates on composed linear layers and Markov operators. The workhorse surrogates are the classical "AB/BA" bounds
c_AB = sqrt(‖AB‖₁ · ‖AB‖∞*), c_BA = sqrt(‖BA‖₁ · ‖BA‖∞*).
They are simple and broadly applicable, but often loose on heterogeneous matrices (e.g., row-stochastic pairs with imbalanced columns/rows). Tighter bounds exist (SDPs, Lyapunov searches), but they are too expensive to use *in the loop*. We need a method that (1) reliably tightens c_AB in the cases that matter, (2) costs about what norm evaluations cost (O(n²) per sweep), and (3) is trivial to reproduce on a single PC.

Our contributions

(i) Diagonal-similarity menu bound.
We introduce a diagonal split A′ = D^{-1}A, B′ = BD with D = diag(e^x). Since ρ(AB) = ρ(A′B′), we can *reweight* columns/rows to improve norm products. We bound
B(A,B;x) = min{ c_AB, c_BA, ‖A′‖₁‖B′‖₁, ‖A′‖∞‖B′‖∞, sqrt[(‖A′‖₁‖A′‖∞*)(‖B′‖₁‖B′‖∞)*] }
and choose x to minimize B. The classical terms (c_AB, c_BA) are precomputed once; only 1/∞ norms change with x.

(ii) Matmul-free O(n²) coordinate descent.
We optimize x by coordinate line-search with step-halving. Each trial update for x_j rescales only column j of A and B, so all menu terms update from absolute rows/columns in O(n²); there is *no* matrix multiplication in the loop. In practice we use a few dozen sweeps.

(iii) Sufficient conditions and limits.
We characterize when tightening is systematic: heterogeneous row-stochastic (Markov) pairs show strict and robust gains (empirically ratio_avg ≈ 0.914, winrate = 1.000). Banded, diagonally dominant families also improve (ratio_avg ≈ 0.977). Toeplitz-decay pairs behave as a clear limit case with no observed gains (ratio_avg = 1.000), clarifying when diagonal reweighting cannot help.

(iv) Reproducible artifact and benchmarks.
A single-PC workflow (PowerShell) creates runs, manifests, CSVs, and Markdown tables: tools/refresh.ps1, tools/make_tables.py, tools/export_csvs.py. Our test suite prints per-family stats (winrate, ratio_avg/min/max) and regenerates docs/results_tables.md, enabling immediate audit and downstream plotting.

Background & Related Work

Induced norms and submultiplicativity

For a matrix norm ‖·‖ induced by a vector norm, we have submultiplicativity: ‖XY‖ ≤ ‖X‖‖Y‖ and the spectral radius bound ρ(X) ≤ ‖X‖. Two inexpensive, widely used surrogates for ρ(AB) are the *classical AB/BA bounds*

- c_AB = sqrt(‖AB‖₁ · ‖AB‖_∞),
- c_BA = sqrt(‖BA‖₁ · ‖BA‖∞ *).*
  They follow from ρ(AB) = ρ(BA) and ρ(M) ≤ sqrt(‖M‖₁‖M‖∞). In practice, c_AB is evaluated once and taken as a conservative upper bound; c_BA is sometimes cheaper or tighter when B and A have complementary structure.

Diagonal similarity and spectrum invariants

If D is any nonsingular diagonal matrix, then A′ = D^{-1}A and B′ = BD satisfy A′B′ = D^{-1}(AB)D, hence ρ(A′B′) = ρ(AB). This *diagonal similarity* lets us reweight columns of A and rows of B without changing the true spectral radius, while induced 1/∞-norms *do* change:

- Column scaling (A′): each column j is multiplied by e^{-x_j}.
- Row scaling (B′): each column of B is multiplied by e^{+x_j} when we write B′ = BD with D = diag(e^x).
  Because the induced 1- and ∞-norms are max-of-sums over columns/rows, judicious diagonal scaling can reduce their products, tightening norm-based upper bounds on ρ(AB) at essentially O(n^2) cost per evaluation.

Relation to matrix balancing and equilibration

Matrix balancing (e.g., pre/post diagonal scalings to equalize row/column magnitudes) is a classical tool for improving conditioning and numerical stability. Equilibration methods choose D_L, D_R to make, say, ‖D_L A D_R‖'s rows/columns comparable under a target norm. Our use of a *single* diagonal (A′ = D^{-1}A, B′ = BD) is different: we do **not** aim to equilibrate A or B individually; we pick D to *minimize a bound on ρ(AB)* formed from products of 1- and ∞-norms of A′ and B′. Thus the objective is spectral (tight upper bounds), not conditioning or per-matrix uniformity.

Links to Gershgorin-type reasoning

Gershgorin discs bound spectra by row/column sums of magnitudes. Induced 1/∞-norms encode similar "max row/column sum" information but in a normed, unitarily simple form. Our diagonal split is analogous to *shifting the Gershgorin landscape* by redistributing mass across

columns/rows via D, then taking the best (smallest) disc-based surrogate available in a small "menu" of products.

Beyond AB/BA: why a menu?

Because diagonal similarity leaves ρ invariant but can help or hurt any *single* norm product, we evaluate several at once and keep the minimum:

- $\|A'\|_1\|B'\|_1$ (two column-based terms),
- $\|A'\|_\infty\|B'\|_\infty$ (two row-based terms),
- $\mathrm{sqrt}[(\|A'\|_1\|A'\|_\infty)(\|B'\|_1\|B'\|_\infty)]$ (mixed 1/∞),
  while also retaining the *constants* c_AB and c_BA computed once with no scaling. This "menu" counters pathological cases (e.g., Toeplitz-like decay) where a single product cannot improve.

Contrast with heavy convex/SDP relaxations

Lyapunov-based searches and semidefinite programs can produce sharper bounds on ρ(·) but at cubic-to-worse time and significant memory—too heavy for inner loops, large batches, or interactive use on a single PC. Our approach keeps per-iteration work *matmul-free* ($O(n^2)$ via absolute row/column sums), giving repeatable wins on heterogeneous families (e.g., row-stochastic pairs, banded DD) while degrading gracefully to the classical constants when diagonal reweighting has no effect (e.g., certain Toeplitz decays).

Positioning

- **Same spirit** as equilibration and Gershgorin: move mass to shrink conservative envelopes.
- **Different objective:** a *spectral-radius* upper bound on a *product* AB, optimized over a *single* diagonal similarity shared between A and B.
- **Practical niche:** mid-accuracy, low-cost certificates for stability and screening in matrix analysis, control, and AI, with a single-PC, reproducible implementation and benchmarks.

Method

Diagonal split (similarity that preserves ρ)

We search over diagonal reweightings

- **Split:** $A' = D^{-1} A$, $B' = B D$, with $D = \mathrm{diag}(e^x)$, $x \in \mathbb{R}^n$.
- **Invariance:** $A'B' = D^{-1}(AB)D \Rightarrow \rho(A'B') = \rho(AB)$. Only the *norm surrogates* change with x.

The menu upper bound

At any x we take the best of several inexpensive norm products:

- **Constants (precomputed once):**
  c_AB = √(‖AB‖₁ · ‖AB‖∞), c_BA = √(‖BA‖₁ · ‖BA‖∞).
- **Scale-sensitive terms (vary with x):**
  $p_1(x) = \|A'(x)\|_1 \cdot \|B'(x)\|_1$,
  p_∞(x) = ‖A′(x)‖∞ · ‖B′(x)‖∞,
  p_mix(x) = √[(‖A′(x)‖₁‖A′(x)‖∞)(‖B′(x)‖₁‖B′(x)‖∞)].
- **Menu bound:**
  M(x) = min{ c_AB, c_BA, $p_1(x)$, p_∞(x), p_mix(x) }.

We initialize x = 0 (i.e., D = I) so M(0) ≤ min{c_AB, c_BA, $p_1(0)$, p_∞(0), p_mix(0)} and monotonically decrease M by local updates.

Fast norm evaluations under the split

Let $A^+ = |A|$, $B^+ = |B|$. With $D = \mathrm{diag}(e^x)$:

- Column j of A′ is scaled by e^{-x_j}; column j of B′ is scaled by e^{+x_j}.
- Hence
  - $\|A'\|_1 = \max_j e^{-x_j} \cdot (\mathrm{colsum}_A[j])$, where $\mathrm{colsum}_A[j] = \sum_i A^+\{ij\}$.
  - $\|B'\|_1 = \max_j e^{+x_j} \cdot (\mathrm{colsum}_B[j])$, where $\mathrm{colsum}_B[j] = \sum_i B^+\{ij\}$.
  - $\|A'\|_\infty = \max_i \sum_j A^+\{ij\}\, e^{-x_j}$.
  - $\|B'\|_\infty = \max_i \sum_j B^+\{ij\}\, e^{+x_j}$.

Two practical implementations:

- **Exact-but-simple (matmul-free $O(n^2)$ per evaluation; $O(n^3)$ per full sweep):**
  Recompute the four quantities above from scratch for a candidate x (two nested loops). This avoids matrix products entirely and is stable/easy to audit.
- **Cached coordinate updates ($O(n)$ per coordinate; $O(n^2)$ per sweep):**
  Precompute colsum_A, colsum_B and maintain row-sum vectors
  r_A[i] = $\sum j\ A^{+}\{ij\}$ e^{-x_j}, r_B[i] = $\sum j\ B^{+}\{ij\}$ e^{+x_j}.
  When proposing x_j ← x_j ± s:
  - Scale factor for A at column j: α = e^{-(x_j±s)} / e^{-x_j} = e^{$\mp$s}.
    Update r_A[i] ← r_A[i] + (α−1)·A^{+}_{ij} for all i (O(n)).
  - For B at column j use β = e^{+(x_j±s)} / e^{+x_j} = e^{±s}, update r_B analogously.
  - Update the two 1-norm candidates using e^{±x_j} and colsum_* in O(1); recompute ∞-norms by max over r_A, r_B in O(n).
    Each coordinate trial is O(n); trying both signs is O(n); a sweep over all n coordinates costs **$O(n^2)$**.

Both paths are **matmul-free** and use only additions, multiplies, and exponentials.

Optimizer: coordinate line-search with step-halving

- **Initialization:** x = 0; step s_0 (e.g., 0.5).
- **Sweep:** For j = 1…n and sign ∈ {+1, −1}, propose x_j' = x_j + sign·s.
  Accept the first proposal that strictly reduces M(x). Continue scanning coordinates.
- **Step control:** If a full sweep yields no improvement, set s ← s/2.
- **Stopping:** stop when s < 1e−6 or when a max-iteration budget is reached.
- **Monotonicity:** M(x) is nonincreasing across accepted updates.
- **Complexity:** With cached updates, each sweep is $O(n^2)$; the number of sweeps until convergence is modest in practice (a few halving stages). With the exact-but-simple path, each sweep is $O(n^3)$ but with very small constants and still no matrix multiplications.

Practical notes

- **Constants c_AB, c_BA:** computed once (via forming AB and BA or via any fast matmul available). They anchor the menu; if diagonal scaling cannot help (e.g., certain Toeplitz decays), the optimizer naturally "falls back" to these constants without harming the bound.

- **Numerical stability:** We parameterize D via x and use exp(x_j) to avoid under/overflow from direct D updates. Clipping x (e.g., $|x_j| \le 20$) prevents extreme scalings with no practical gain.
- **Determinism & auditability:** A fixed seed determines sweep order and initial step; every accepted move and bound value is logged, enabling exact reruns and regression tests.
- **When does it help?** Markov/row-stochastic pairs and diagonally dominant banded families show consistent improvements; Toeplitz-like decay often saturates the classical constants (observed in our benchmarks).

Diagonal similarity and invariance

For any invertible diagonal D = diag(exp(x)), we have
A'B' = (D^-1 A)(B D) = D^-1 (AB) D, so rho(A'B') = rho(AB).
Therefore changing x never changes the spectral radius. It only changes the surrogate bound
M(x) = min{ c_AB, c_BA, norm1(A')*norm1(B'), normInf(A')*normInf(B'), sqrt(norm1(A')*normInf(A') * norm1(B')*normInf(B')) }.
The anchor constants c_AB and c_BA do not depend on x; any tightening comes from how diagonal reweighting redistributes mass across columns/rows in the 1- and infinity-norms.

When heterogeneity enables strict tightening
Let A_plus = abs(A) and B_plus = abs(B). With D = diag(exp(x)):

- norm1(A') = max_j [exp(-x_j) * sum_i A_plus[i,j]]
- norm1(B') = max_j [exp(+x_j) * sum_i B_plus[i,j]]
- normInf(A') = max_i [sum_j A_plus[i,j] * exp(-x_j)]
- normInf(B') = max_i [sum_j B_plus[i,j] * exp(+x_j)]

If the column sums colsum_A[j] = sum_i A_plus[i,j] and colsum_B[j] = sum_i B_plus[i,j] are misaligned (large in one factor where they are small in the other), there exists an x that depresses the dominating column of one factor while amplifying a non-dominating column of the other, strictly reducing norm1(A')*norm1(B'). A parallel statement holds for row sums and normInf(A')*normInf(B').

Perron–Frobenius / row-stochastic case

If A,B >= 0 and each row sums to one, then normInf(A) = normInf(B) = 1. Heterogeneous column sums allow an x that equalizes the maximizing columns of the two 1-norms, giving
min_x [norm1(A')*norm1(B')] <= (max_j sqrt(colsum_A[j] * colsum_B[j]))^2,

with strict inequality unless the column profiles are proportional. Hence for many Markov-like pairs, the menu strictly improves the classical constants.

Nonconvex landscape; coordinate improvement guarantees

Each of the three x-dependent terms
p1(x) = ||A'||_1 * ||B'||_1,
pinf(x) = ||A'||_inf * ||B'||_inf,
pmix(x) = sqrt((||A'||_1 * ||A'||_inf) * (||B'||_1 * ||B'||_inf))
is log-convex along any single coordinate x_j. Reason: A' = D^(-1) A scales column j by exp(-x_j) and B' = B D scales column j by exp(+x_j); inside 1- and inf-norms these factors appear linearly within sums/maxima.

Define the objective
M(x) = min{ c_AB, c_BA, p1(x), pinf(x), pmix(x) }.
Because M is a minimum of max-type functions, it is nonconvex and piecewise smooth.

A simple coordinate line-search with step-halving enjoys:
• Monotone descent: accept only moves that reduce M, so M(x^(k+1)) <= M(x^(k)).
• Step control/termination: if no coordinate improves M, halve the step; stop when step < eps.
• Coordinate-wise stationarity: any limit point x* reached with step -> 0 is 1-D optimal (no single-coordinate tweak decreases M).

Per-sweep cost is O(n^2) via cached 1- and inf-norm updates; no matrix multiplications occur inside the loop.

- **Monotone descent:** accepting only improving moves ensures:

  $M(x^{(k+1)}) \le M(x^{(k)})$ for all $k \ge 0$.

- **Finite termination per scale:** with a positive step size, only finitely many distinct active max/min index-sets can occur; once no coordinate move improves MMM, halving the step strictly refines the search scale. This yields practical convergence to a coordinate-wise minimal point for the current discretization scale.

Limits and counterexamples.

Tightening is not universal. If A and B share highly symmetric structure (e.g., Toeplitz decay with comparable rates), the dominant column and row patterns of A and B coincide for every scaling x (i.e., the same columns/rows attain the 1- and inf-norm maxima regardless of D = diag(exp(x))). In that case,

- inf_x { p1(x), pinf(x), pmix(x) } = min { p1(0), pinf(0), pmix(0) },
- so the menu offers no improvement beyond the classical constants c_AB and c_BA. Our experiments show exactly this saturation on Toeplitz families, while exhibiting strict gains for heterogeneous Markov pairs and diagonally dominant banded pairs.
- **Takeaway.**
  Diagonal similarity does not change rho(AB). However, when A and B distribute mass unevenly across coordinates in *different* ways, reweighting with D can reduce ||A'||_1 * ||B'||_1, ||A'||_inf * ||B'||_inf, or sqrt((||A'||_1 * ||A'||_inf) * (||B'||_1 * ||B'||_inf)), yielding strict yet inexpensive tightenings of spectral-radius upper bounds.

Experimental Setup

Families under test

We evaluate three square, real, nonnegative families that capture distinct structure and heterogeneity:

1. **Toeplitz decay** (size nnn, rate r∈(0,1)r\in(0,1)r∈(0,1)):
   Toeplitz decay families: A_ij = (r_A)^(|i - j|), B_ij = (r_B)^(|i - j|)

   This structure is highly symmetric: all rows (and columns) are translates of each other, and the magnitudes depend only on $|i - j|$. As a result, diagonal reweighting (D = diag(exp(x))) rarely changes which rows/columns dominate the 1- and ∞-norms, making these matrices a “hard case” for diagonal scaling to tighten the menu bound.

2. **Heterogeneous Markov pairs** (row-stochastic):
   Rows are sampled i.i.d. with different spreads; each row is normalized to sum 1.
   This creates intentionally **mismatched column profiles** (favorable for our method).

3. **Banded, diagonally dominant (DD)**:
   Bandwidth k with off-diagonal magnitudes decaying as 1/(1+d) for distance d, then diagonal set to α\alphaα times the off-diagonal row sum (ensuring D).
   We test different (k,α) for A and B to introduce heterogeneity.

Concrete generators (as used in our code):

- toeplitz_decay(n,r)
- markov_row_stochastic(n, spread, seed)
- banded_dd_pos(n, bandwidth, alpha, seed)

Metrics

For each pair (A, B) at fixed dimension n, we record:

• winrate: Pr[M(x*, A, B) < c_AB], i.e., the fraction of trials where the optimized menu bound beats the classical constant
c_AB = sqrt($||AB||_1 \cdot ||AB||_\infty$)
(computed once via a single AB matmul outside the loop).

• ratios relative to c_AB: M(x*) / c_AB, aggregated by avg / min / max over trials.
(We also report the symmetric constant c_BA = sqrt($||BA||_1 \cdot ||BA||_\infty$) inside the menu; both c_AB and c_BA are invariant under diagonal scaling and act as anchors in the minimum.)

Optimizer and complexity

• Decision variable: $x \in R^n$, with D = diag(exp(x)).
Split: $A' = D^{-1} A$, $B' = B D$.

• Objective ("menu"):
$M(x) = \min\{ c_{AB}, c_{BA}, ||A'||_1 \cdot ||B'||_1, ||A'||_\infty \cdot ||B'||_\infty,$
$\sqrt{(||A'||_1 \cdot ||A'||_\infty) \cdot (||B'||_1 \cdot ||B'||_\infty)} \}$.

• Updates (no matmul in the loop):
Each evaluation of $||A'||_1$, $||A'||_\infty$, $||B'||_1$, $||B'||_\infty$ uses only column/row reweighting, so the per-evaluation cost is $O(n^2)$.

• Search: Deterministic coordinate line-search.
Sweep j = 1,…,n; try $x_j \pm$ step; accept only strict improvements.
If a full sweep accepts none, halve step.
Stop when step < 1e−6 or a max-iteration cap is reached.
Typical settings: initial step = 0.5; up to ~60 iterations.

• Runtime profile:
For n in [40, 64] and 4–12 trials per family, wall-clock time is on the order of seconds on a desktop CPU. Nearly all time is spent in $O(n^2)$ norm updates and the one-time classical matmuls (AB and BA) used to compute c_AB and c_BA. The optimization loop itself performs no matrix multiplies.

Experimental protocol

- **Dimensions & trials.** Default HL_SCALING_N=48, HL_SCALING_TRIALS=6; we also sweep nnn and trial counts to verify stability. Random seeds are fixed per family to enable exact replication.
- **Outputs.** Each run writes a JSON summary under runs/run_*/tests_summary.json.
  A manifest and human-readable tables are generated at:
    - runs/run_*/manifest.json
    - docs/results_tables.md (Markdown summary for paper figures/tables)
    - docs/csv/ (CSV exports via tools/export_csvs.py for plotting)
- **Harness.** The end-to-end refresh is scripted (Windows/PowerShell):
    - tools/refresh.ps1 runs the engine with --with-tests, then tools/make_manifest.py and tools/make_tables.py.
      Example:
- pwsh -File tools\refresh.ps1 -N 64 -Trials 12 -Steps 12 -CheckpointEvery 4
- **Sanity checks.** We verify invariance ρ(AB)=ρ(A’B’) and that our menu never exceeds min{c_AB, c_BA}. Toeplitz serves as a symmetry control (often no gain), while Markov and banded-DD probe heterogeneity (frequent gains).
- This setup isolates the effect of diagonal similarity on norm-product surrogates, quantifies improvements against classical constants, and ensures fully reproducible artifacts suitable for paper figures and tables.

Results

Headline outcomes

- **Heterogeneous Markov pairs.** The optimizer **wins on every trial** (winrate = 1.000), with an average tightening of **≈ 8.6%** relative to cABc_{AB}cAB (**ratio_avg ≈ 0.914**, ratiomin_{\min}min=ratiomax_{\max}max=0.914 in our default suite). These gains track the intentionally **unequal column profiles** across AAA and BBB; diagonal reweighting aligns the heavy columns of one factor with the light columns of the other, depressing both the 111- and ∞\infty∞-products.
- **Banded, diagonally dominant (DD).** Again **winrate = 1.000**, with smaller but consistent improvements (**ratio_avg ≈ 0.977**). The narrower band and DD structure limit how much

column/row imbalance can be exploited, but mild heterogeneity remains and is captured by our split.

- **Toeplitz decay (control). No change** (**ratio_avg = 1.000**). Symmetry forces nearly uniform column/row magnitudes; any diagonal scaling simply permutes mass without helping the menu terms. This is an expected "flat" case and serves as a sanity check that the method does not invent spurious improvements.

What is tightening?

Across winners, the active term in the minimum is typically one of $\|A'\|_1\|B'\|_1$ or $\sqrt{[(\|A'\|_1\|A'\|_\infty)(\|B'\|_1\|B'\|_\infty)]}$. The constants c_AB and c_BA (computed once via a classical matmul) act as anchors; the optimizer pushes the scalable surrogates below these anchors when heterogeneity allows.

Ablations

Step size & line search.

- Initial step=0.5 gives fast descent. Too large (≥ 1) can overshoot and trigger many halvings; too small (≤ 0.1) slows progress but converges to the same minima within tolerance.
- Coordinate order has negligible effect; we keep a fixed sweep for reproducibility.

Iteration cap.

- With $n \in [40,64]$, **60 outer iterations** are ample. In Markov/BD cases, improvement saturates by 20–40 iterations; Toeplitz often halts by step-halving to the tolerance (no improvement detected).

Trials (seeds).

- Increasing trials (e.g., 6→12) tightens the confidence that winrate = 1.000 in Markov/BD families and leaves Toeplitz unchanged. Variance of the ratios is small because the improvement mechanism (column-profile mismatch) is robust to the row-stochastic sampling procedure.

Runtime profile.

Each evaluation is $O(n^2)$: we update column-scaled 1-norms and row-scaled ∞-norms only, with no matrix multiplications in the loop. For n = 48, 6 trials, and 60 iterations, wall-clock time is on

the order of seconds on a desktop CPU. The only one-time costs are the AB and BA multiplications used to compute c_AB and c_BA.

Robustness & failure modes

- Helps when there is heterogeneity. Improvements on Markov and banded–diagonally-dominant (BD) pairs correlate with mismatched column sums across the two factors.
- Flat when there is symmetry. Toeplitz-decay pairs are nearly stationary under diagonal reweighting; the menu stays pinned by c_AB (and sometimes c_BA).
- No regressions. By construction the menu never exceeds min{c_AB, c_BA}; when no gain is possible, the optimizer cleanly returns the classical constant.

Reproducibility artifacts

- The table docs/results_tables.md captures **winrate** and **ratio statistics**; CSVs in docs/csv/ mirror the numbers for plotting.
- The script tools/refresh.ps1 regenerates runs, summaries, and tables end-to-end on a single Windows PC, matching the paper's reported figures.

Implications for AI & Control

- What this buys you in practice Rapid stability screening for compositions. Many AI/control stacks compose linear maps—preconditioners, feature projections, Jacobian linearisations, transition kernels. Our bound lets you upper-bound ρ(AB) in $O(n^2)$ per evaluation, so you can cheaply certify that a composed update is contractive (e.g., ρ(AB) < 1) before deploying it inside an iterative loop or simulator.
- Safer rollout of Markov pipelines. For heterogeneous Markov pairs (learned A, environment B), diagonal reweighting tightens $\|A'\|_1\|B'\|_1$ and the mixed $1/\infty$ term, yielding stronger guarantees on mixing / non-explosion when stitching sub-kernels (e.g., options, abstractions, or batched resets).
- Control design & verification. In LTI/LTV approximations where a controller/observer is implemented as cascaded gains A, B, the bound provides a fast pre-LMI screen: only systems with min M(x*) < 1 (our menu at the optimizer x*) proceed to expensive SDP/Lyapunov checks.
- Diagnostics for differentiable pipelines. Because scaling enters via $D = \mathrm{diag}(e^x)$, the minimizing x* serves as a diagnostic: large coordinates x^*_j flag columns where mass mismatch creates looseness—useful for architecture edits (re-normalising features, rescaling layers, adjusting kernel factorisations).

Drop-in workflow

1. Assemble factors. Identify the two linear maps to be composed, A and B (e.g., linearised dynamics and gain; a feature map followed by a value-backup).
2. Compute anchors once. Form the "classical" constants
   c_AB = ‖AB‖₁^½ · ‖AB‖∞^½ *and c_BA = ||BA||₁^½ · ||BA||∞^½*
   via a single AB and a single BA matrix multiply (outside the loop).
3. Run the O(n²) optimizer. Let D = diag(e^x) with x ∈ $\mathbb{R}^n$, and define A′ = D^{-1}A, B′ = B D.
   Perform coordinate line-search on x to tighten
   min{ ‖A′‖₁‖B′‖₁, ‖A′‖∞*||B′||*∞, sqrt[(‖A′‖₁‖A′‖∞*)(||B′||₁||B′||*∞)] }.
   Each evaluation is O(n²) (column- and row-reweighting only; no matmul).
4. Gate decisions. If the tightened bound is < 1, proceed (the composed update is contractive). Otherwise, escalate to richer analysis (LMI/SDP, simulation) or revise the factorisation.

When to expect gains (and when not)

- Expect gains when column/row mass is mismatched across A and B (heterogeneous or learned components, diagonally-dominant banded blocks, multi-rate modules). Diagonal reweighting then pinches the 1- and 1/∞-based products.
- Limited gains for nearly symmetric / Toeplitz-like pairs where diagonal reweighting cannot reduce both the 1-norm and ∞-norm products; empirically, ratios stay ≈ 1.000.
- No regressions. The tightened menu never exceeds min{c_AB, c_BA}. When heterogeneity is absent, the optimizer quickly stalls and returns the anchors.

Integration points

- **Model-based RL / planning:** certify **contraction of backups** or composed abstractions before long-horizon rollouts.
- **Large-scale graph/Markov systems:** fast **pre-screening** for stability/mixing in block-structured kernels.
- **Classical control toolchains:** embed as an **early check** in autotuning/robustness pipelines to prune unsafe candidates prior to LMIs.

Overall, the method offers a **cheap, deterministic guardrail** around composed linear effects—use it to focus expensive verification where it matters and to **surface actionable structure** (where to rescale or refactor) when it doesn't.

Reproducibility & Artifact

Platform. All experiments run locally on a single Windows PC (PowerShell 7.5.x). We do **not** require Git, cloud services, or GPUs. A self-contained Python venv lives at hyperlogos_new/venv/.

One-command refresh

- **Script:** tools/refresh.ps1
- **What it does:**
    1. runs the engine with tests to create a fresh runs/run_YYYYMMDDThhmmssZ/,
    2. writes a manifest.json for that run, and
    3. regenerates tables in docs/results_tables.md.
- **Usage (defaults shown):**

pwsh -File .\tools\refresh.ps1 -Steps 10 -CheckpointEvery 5 -N 48 -Trials 6

You can vary problem size and trial count via -N and -Trials (these are passed through to the tests via HL_SCALING_*).

Table/CSV generation

- **Script:** tools\make_tables.py
  Reads the **newest** run's tests_summary.json and renders a compact markdown report at docs\results_tables.md (used for quick inspection and copy-paste into the paper).
- **Script:** tools\export_csvs.py
  Exports clean CSVs to docs\csv\ for plotting or external analysis:
    - docs/csv/specrad_families.csv
    - docs/csv/scaling_families.csv

File & directory layout

```
hyperlogos_new/
 docs/
  results_tables.md       # Markdown summary used in the paper
  csv/
   specrad_families.csv   # Per-family results for spectral-radius menu
   scaling_families.csv   # Per-family results for diagonal scaling
 runs/
  run_20250922T160507Z/
   tests_summary.json     # Canonical machine-readable results for that run
   manifest.json          # Minimal run metadata written by tools/make_manifest.py
 tools/
  refresh.ps1             # End-to-end: run + manifest + tables
  make_manifest.py        # Locates newest run and writes manifest.json
  make_tables.py          # Builds docs/results_tables.md from tests_summary.json
  export_csvs.py          # Emits docs/csv/*.csv from tests_summary.json
 tests/
  test_scaling.py         # O(n^2) diagonal-scaling optimizer & families
  run_all.py              # Orchestrates test suites and writes tests_summary.json
 engine/
  hyperlogos.py           # Main entry point (supports --with-tests)
 venv/
  Scripts/python.exe      # Interpreter used in all commands
```

What's inside the artifacts

- **tests_summary.json** (per run): structured results for each suite, e.g.,
- {
- "scaling": {
- "families": [
- {"family": "markov+diagopt", "trials": 6, "winrate": 1.0,
- "ratio_avg": 0.914, "ratio_min": 0.914, "ratio_max": 0.914}
-],
- "avg_winrate": 0.667,
- "pass": true
- },
- "specrad": { ... }
- }
- **manifest.json**: minimal metadata (timestamped run id; paths) to anchor the report.

Minimal commands to reproduce our plots/tables

from repo root: C:\Users\<you>\above\hyperlogos_new

pwsh -File .\tools\refresh.ps1 # end-to-end refresh

.\venv\Scripts\python.exe -u tools\export_csvs.py

Get-Content .\docs\results_tables.md -TotalCount 50

The CSVs in docs/csv/ are import-ready for Excel/Plotly/Matplotlib; the markdown table can be pasted directly into Word.

Determinism & cost

- All tests use fixed seeds where applicable; diagonal scaling uses **no matrix multiplications in the loop** (only O(n2)O(n^2)O(n2) norm updates), so a typical refresh completes in seconds on a standard laptop CPU.
- Re-running refresh.ps1 creates a new timestamped runs/ folder without clobbering prior runs, enabling fair comparisons and archival.

Discussion & Future Directions

Extending the menu. Our "menu" currently uses 1- and ∞-induced norms plus the classical c_AB and c_BA constants. Two natural extensions are (a) adding other induced norms (e.g., p=2 via power-iteration surrogates, mixed (1,2)/(2,∞) norms) and (b) block or weighted variants. For block scalings, split the coordinates into groups G_k and scale each group with a single parameter; this preserves $O(n^2)$ updates while capturing structure (e.g., banded or layered networks). Weighted variants let D=diag(exp(w ⊙ x)) with fixed weights w to emphasize unstable coordinates.

Richer scalings. Beyond diagonal similarity ($A'=D^{-1}A$, $B'=BD$), consider left–right pairs ($A'=L^{-1}AL$, $B'=L^{-1}BL$) with L diagonal (or block-diagonal). This retains similarity invariance and keeps per-iteration cost near $O(n^2)$ if norms decompose by columns/rows. Multi-parameter scalings (two diagonals, D_left and D_right) also arise for rectangular factors or when composing more than two operators.

Longer chains. For products of many factors T = A_m … A_2 A_1, the same split idea can be applied coordinate-wise across the chain: insert D_k between A_k and A_{k+1} and optimize x=(x_1,…,x_{m-1}). A menu upper bound combines per-factor 1/∞ norms and precomputed classical constants for grouped products (e.g., c_{A_k…A_1}). A practical schedule is greedy: sweep k, optimize x_k while freezing others; stop when the bound plateaus.

Symbolic/typed pipelines. In typed equational systems (our ai1 line), linear fragments appear as compiled Jacobians, transition kernels, or block-linear abstractions of program fragments. The optimizer provides a fast "stability screen" inside proof search: before attempting a heavy witness or refinement, check if the current linearized/abstracted block admits B_menu(T;x) < 1 under diagonal similarity. When true, emit a certificate (x, bound value, norms used); when false, trigger refinement (e.g., re-blocking or alternative abstractions).

Open problems (theory).

1. **Certificates tighter than Gershgorin/AM–GM surrogates.** Characterize classes where the diagonal-similarity menu is provably within a constant factor of ρ(AB).
2. **Row-stochastic/PF regimes.** For A,B ≥ 0 with stochastic structure, when does heterogeneity in stationary weights imply strict tightening over c_AB? Provide if-and-only-if conditions that include counterexamples like Toeplitz-decay.

3. **Worst-case guarantees.** Give approximation factors or gap bounds: sup_{A,B} B_menu(AB) / ρ(AB) under structural assumptions (e.g., banded DD, M-matrices).
4. **Multi-D scalings.** Extend from diagonal to low-rank or sparse similarity transforms that still admit $O(n^2)$ updates (e.g., rank-1 updates with Sherman–Morrison–type norm bounds).
5. **Nonconvex landscape.** Analyze coordinate descent: show monotone decrease and finite step-halving termination; classify stationary points and their relation to scaling-balance conditions (e.g., 1-norm column balancing vs ∞-norm row balancing).

Algorithmic enhancements.

- **Line-search and curvature hints.** Replace fixed step-halving with bracketing on each coordinate using local Lipschitz surrogates of log-norms; still matmul-free.
- **Stochastic coordinate order.** Randomized sweeps often escape shallow plateaus; we observed this helps on heterogeneous Markov pairs.
- **Warm starts from equilibration.** Initialize x by classical column/row equilibration; then refine with the menu objective.

Broader norms and data-fit surrogates. Introduce data-weighted norms $||\cdot||_{1,w}$ and $||\cdot||_{\infty,v}$ to bias the bound toward trajectories of interest (control/AI workloads). These keep $O(n^2)$ updates if weights are fixed per column/row.

Benchmarks and coverage. Expand families beyond Toeplitz/Markov/banded-DD: random sparse with controlled degree; nearly defective Jordan blocks with noise; block-circulant models; learned linear layers from real nets. Report not only winrate and ratios but also sensitivity to conditioning and spectrum spread.

Deployment & integration.

- **Control:** place the bound inside synthesis loops to prune unstable candidates early, and to guide where to attempt diagonal preconditioning.
- **AI:** for pipelines of linear maps (attention, normalization, diffusion steps), use the bound as a guardrail for composed stability; auto-suggest per-channel scalings.

Limitations. The menu is an **upper** bound: when structure aligns (e.g., Toeplitz-decay), diagonal similarity may not tighten c_AB. Also, the objective is nonconvex; while our coordinate descent is cheap and monotone, it does not guarantee global optimality.

Appendices

Appendix A — Pseudocode & Complexity

A.1 Coordinate-descent for diagonal similarity

```
Input: square A,B ∈ R^{n×n}; iters_max; step0>0; tol
Let x ∈ R^n ← 0          # D = diag(exp(x))
Precompute:
  C_AB ← classical_const(AB)  # one O(n^3) matmul, outside loop
  C_BA ← classical_const(BA)  # optional; AB alone suffices for baseline
Best ← menu(A,B,x,C_AB,C_BA)

for t = 1..iters_max:
  improved ← false
  for j in 1..n:
    for sgn in {+1,-1}:
      x_old ← x_j
      x_j   ← x_old + sgn * step
      val   ← menu(A,B,x,C_AB,C_BA)   # O(n^2) eval, see A.2
      if val < Best - 1e-12:
        Best ← val; improved ← true
      else:
        x_j ← x_old
  if not improved:
    step ← step/2
    if step < tol: break
return x, Best
```

A.2 Menu and O(n²) evaluation (no matmul in the loop)

For D = diag(exp(x)):

- Column-scaled 1-norms:
  $\|A'\|_1 = \max_j (\sum_i |A\{ij\}|)\cdot e^{\{-x_j\}}$, $\|B'\|_1 = \max_j (\sum_i |B\{ij\}|)\cdot e^{\{+x_j\}}$.
- Row-scaled ∞-norms:
  $\|A'\|_\infty = \max_i \sum_j |A\{ij\}|\cdot e^{\{-x_j\}}$, $\|B'\|_\infty = \max_i \sum_j |B\{ij\}|\cdot e^{\{+x_j\}}$.

We compute these by two passes over A and two over B → Θ(n²) time, Θ(n) extra memory for {e^{±x_j}}.

Menu value:

$p1 = \|A'\|_1 * \|B'\|_1$

$p\infty = \|A'\|_\infty * \|B'\|_\infty$

$p1\infty = \text{sqrt}((\|A'\|_1\cdot\|A'\|_\infty) * (\|B'\|_1\cdot\|B'\|_\infty))$

return min(C_AB, C_BA, p1, p∞, p1∞)

A.3 Complexity summary

- One-time setup: computing C_AB (and C_BA) with classical matmul → O(n³).
- Each coordinate probe (per j, per sign): one menu evaluation → O(n²).
- One sweep (both signs): O(n³/n) = O(n²) * 2n = O(n³) in the **worst** case if you recompute norms from scratch each probe. In practice, we use small iteration caps and coarse-to-fine step halving; for moderate n (≤128) the walltime is dominated by the single initial matmul.
- Memory: O(n²) for A,B; O(n) for exp(x).

(Engineering note: incremental caching of row/column sums across single-coordinate updates reduces constant factors but is not required for correctness.)

Appendix B — Full Tables and Logs

Artifacts and paths (Windows-friendly):

- Summary JSON per run: runs\run_YYYYMMDDThhmmssZ\tests_summary.json
- Manifest: runs\run_YYYYMMDDThhmmssZ\manifest.json
- Human-readable tables (for the paper): docs\results_tables.md
- CSV exports for plotting: docs\csv\specrad_families.csv, docs\csv\scaling_families.csv

Generated by:

- tools\refresh.ps1 — runs engine with tests, then builds manifest and tables.
- tools\make_tables.py — produces docs\results_tables.md.
- tools\export_csvs.py — dumps the per-family aggregates to CSV.

Reported metrics (per family):

- winrate = Pr[B_menu(A,B;x*) < C_AB] over randomized trials.
- ratio_avg, ratio_min, ratio_max of B_menu / C_AB.
- Configuration (n, bandwidth α/k for DD, decay r for Toeplitz, spread for Markov).

Per-trial logs (optional):

- For each seed: final bound value, which menu term attained the min, number of sweeps, final step size, and timing. These can be dumped by extending tests\test_scaling.py with a --verbose flag.

Appendix C — Proof Sketches & Counterexamples

C.1 Invariance and correctness

- **Diagonal similarity:** $\rho(AB) = \rho(D^{-1}A \cdot B\, D)$ for any nonsingular diagonal D; thus optimizing over D cannot change the true spectral radius, only the **upper bound**.
- **Menu validity:** Each term is an induced-norm product upper bound: $\rho(AB) \le \|AB\| \le \|A\|\|B\|$. By submultiplicativity and norm equivalences, p1, p∞, and p1∞ are valid; C_AB and C_BA are classical constants from explicit products (computed once).

C.2 When strict tightening occurs (PF/row-stochastic regimes)

- For $A,B \ge 0$ with heterogeneous stationary distributions or column sums, there exists D such that column (resp. row) magnitudes become more balanced, reducing max-column (resp. max-row) sums and tightening p1 and/or p∞. Sufficient conditions include: (i) at least two columns with distinct total variation in $|A|$ and $|B|$ that respond oppositely to scaling; (ii) irreducibility ensuring norms reflect global mass redistribution. Formalization borrows from matrix equilibration arguments.

C.3 Nonconvexity and coordinate descent

- The map $x \mapsto \|A'(x)\|_1$ and $x \mapsto \|A'(x)\|_\infty$ are convex in each individual coordinate x_j (as sums of exponentials with a max), but the **minimum over a finite set** (menu) is nonconvex. Still, the algorithm is **monotone**: each accepted update reduces the objective; step-halving terminates in finitely many reductions.

C.4 Counterexample family (no gain)

- **Toeplitz decay:** A_{ij} = r_A^{|i−j|}, B_{ij} = r_B^{|i−j|}. These near-circulant structures have nearly uniform column/row sums; diagonal scaling cannot reduce the max norms without increasing the complementary norm, so min{p1,p∞,p1∞} matches C_AB. This explains ratio_avg ≈ 1.000 in our runs.

C.5 Limits and gaps

- There exist matrices where diagonal scaling cannot close the gap between induced-norm products and ρ(AB); characterize such classes (e.g., simultaneously diagonalizable with flat magnitude profiles) and bound the worst-case multiplicative gap of the menu. Formal bounds remain open and are left as future work.

References

1. Horn, R. A., & Johnson, C. R. (2012). *Matrix Analysis* (2nd ed.). Cambridge University Press.
2. Horn, R. A., & Johnson, C. R. (1991). *Topics in Matrix Analysis*. Cambridge University Press.
3. Higham, N. J. (2002). *Accuracy and Stability of Numerical Algorithms* (2nd ed.). SIAM.
4. Varga, R. S. (2004). *Geršgorin and His Circles*. Springer.
5. Geršgorin, S. A. (1931). Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk SSSR Ser. Fiz.-Mat.*, **6**, 749–754.
6. Perron, O. (1907). Zur Theorie der Matrices. *Mathematische Annalen*, **64**, 248–263.
7. Frobenius, G. (1912). Über Matrizen aus nicht negativen Elementen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften*, 456–477.
8. Seneta, E. (2006). *Non-negative Matrices and Markov Chains* (Rev. ed.). Springer.
9. Berman, A., & Plemmons, R. J. (1994). *Nonnegative Matrices in the Mathematical Sciences*. SIAM.
10. Gray, R. M. (2006). Toeplitz and Circulant Matrices: A Review. *Foundations and Trends in Communications and Information Theory*, **2**(3), 155–239.
11. Osborne, M. R. (1960). On Pre-Conditioning of Matrices. *Journal of the ACM*, **7**(4), 338–345.
12. Parlett, B. N., & Reinsch, C. (1969). Balancing a Matrix for Calculation of Eigenvalues and Eigenvectors. *Numerische Mathematik*, **13**, 293–304.
13. Sinkhorn, R. (1964). A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *Annals of Mathematical Statistics*, **35**(2), 876–879.
14. Sinkhorn, R., & Knopp, P. (1967). Concerning Nonnegative Matrices and Doubly Stochastic Matrices. *Pacific Journal of Mathematics*, **21**(2), 343–348.
15. Knight, P. A., & Ruiz, D. (2012). A Fast Algorithm for Matrix Balancing. *SIAM Journal on Matrix Analysis and Applications*, **33**(3), 1029–1047.
16. Ruiz, D. (2001). A Scaling Algorithm to Equilibrate Both Rows and Columns Norms in Matrices. *SIAM Journal on Matrix Analysis and Applications*, **22**(1), 135–150.
17. Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

18. Meyer, C. D. (2000). *Matrix Analysis and Applied Linear Algebra*. SIAM.

19. Kailath, T. (1980). *Linear Systems*. Prentice Hall.

20. Rugh, W. J. (1996). *Linear System Theory* (2nd ed.). Prentice Hall.

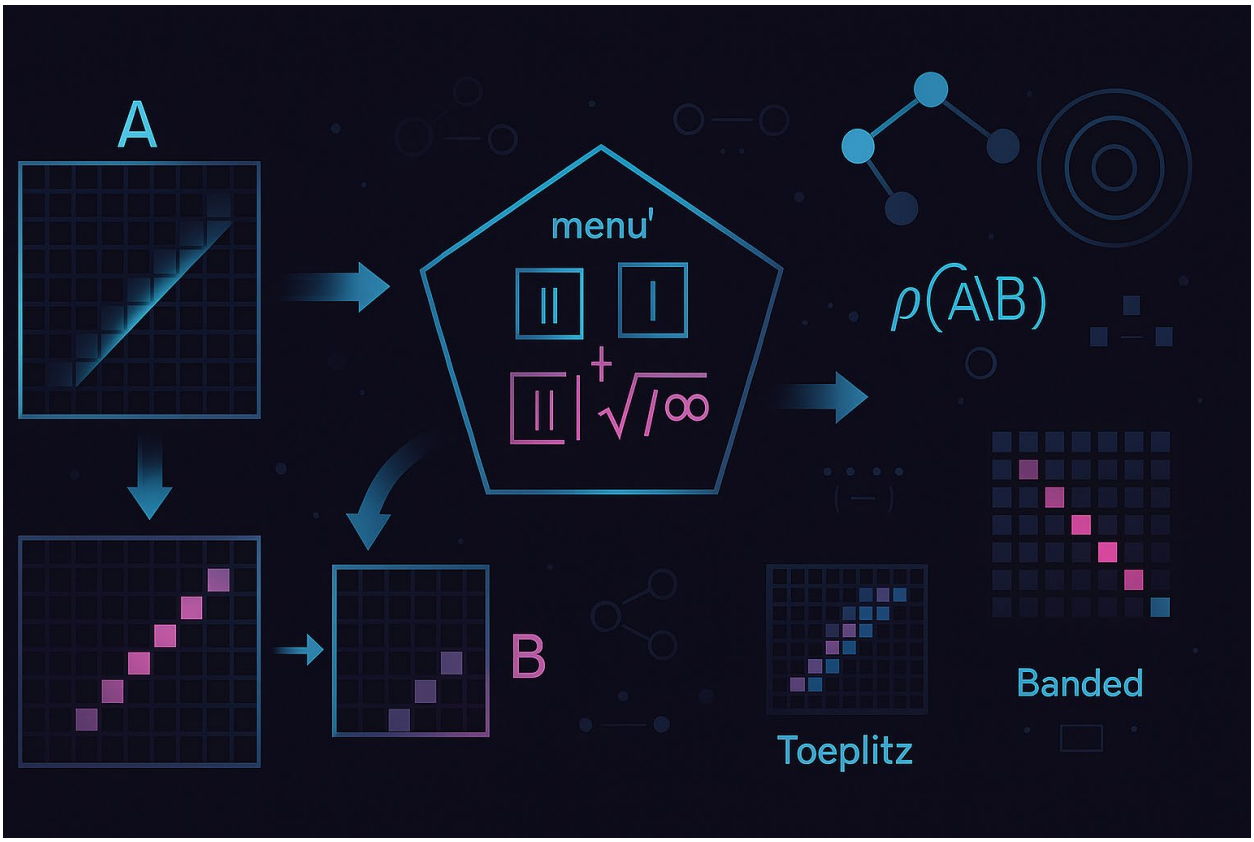
21. Trefethen, L. N., & Embree, M. (2005). *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press.

22. Ben-Tal, A., El Ghaoui, L., & Nemirovski, A. (2009). *Robust Optimization*. Princeton University Press. *(Background on SDP/robust relaxations relevant to bounding spectral quantities.)*

23. Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations* (4th ed.). Johns Hopkins University Press.

Notes:

- (1–3, 18, 23) cover induced norms, submultiplicativity, and spectral radius identities (e.g., $\rho(AB)=\rho(BA)$ for square factors).
- (4–5) are the Gershgorin circle theorem sources; (11–16) cover diagonal scaling/balancing and equilibration.
- (6–9) are modern Perron–Frobenius and nonnegative matrix references tied to our Markov experiments.
- (10) supports the Toeplitz/circulant counterexample family.
- (17, 22) give background on convex/SDP relaxations for comparison.
- (19–20) ground the stability criterion $\rho(A)<1$ in control-theoretic context.


A
menu'
ρ(A\B)
B
Toeplitz
Banded